IN THE SPECIFICATION:

Please amend the specification, as shown on the following pages.

At page 2, line 14, change the paragraph to read as follows:

The Component Balancer described herein establishes and maintains response time goals for selected business logic contained in methods in component-based applications. The premise for this optimization is that business logic in component methods can be driven to response-time goals by selectively delaying business logic in other component methods - e.g. two methods accessing the same table in a database.

At page 2, line 25, through page 3 line 1, change the paragraph to read as follows:

The Component Balancer leverages infrastructure technologies that support component-based optimizations, allowing the Component Balancer to work on a running system without recompiling or redeploying the application. The Component Balancer establishes and maintains response-time goals on business logic contained in methods in component technologies - such as COM+, .NET and BEA Weblogic EJB. As the load on the system varies, the Component Balancer attempts to keep methods with response-time objectives at a specified target performance level at the expense of lower priority methods.

At page 5, lines 8-13, change the paragraph to read as follows:

The Component Balancer establishes and maintains response-time goals on "business logic"- such as COM+, .NET and WLS methods. It does this by delaying other, related business processes that can occur in different applications\application servers or on different machines. As the "load" on the system increases, the business processes with response-time goals attempt to remain at a user specified target performance level. As the load decreases, the delays on the other business processes are reduced.

At page 13, line 14, change the paragraph to read as follows:

The individual methods in the optimization groups are graphically monitored against the goal. Running average response times in milliseconds are plotted against the response-time goal and running average calls per second are also graphed to show load trending.

At page 14, line 26, change the paragraph to read as follows:

One embodiment of the invention is a method that establishes and maintains response-time goals for selected business logic contained in methods used in component-based applications. Benefits of the described Component Balancer are high priority methods that respond consistently under variable load, and provide smoother server resource utilization under peak load. The key features of the Component Balancer include an analysis capability that provides optimization hints, the ability to self-tune under variable load across machines, applications and application servers, and the option of an automated or manual operation.

At page 15, lines 15-18 , change the paragraph to read as follows:

Since the conditioning is added to a running system, infrastructure and tooling is necessary to manage the conditioning lifecycle of discovering and capturing, organizing and conditioning components and their methods. This infrastructure and tooling is called the Component Runtime Conditioner (CRC). A graphical user interface (GUI) is provided to do this that is easy to use (task oriented, drag-and-drop conditioning) and has a persistent store to remember what conditioners have been applied to component's methods on which particular machines and also their current status ----- running, scheduled or stopped. This was described in the cited USSN 10/626,222 in the Cross-References.

At page 15, line 29, change the paragraph to read as follows:

The major functions of the Component Balancer are: (a) to discover and capture applications, (b) to analyze and configure optimization (manually or automatically) and (c) to monitor response-time objectives.

At page 19, line 19, change the paragraph to read as follows:

The Component Balancer automation function 1004 does automatically what a user would do manually. The Component Runtime Conditioner (CRC) and Component Balancer components are capable of recovering from management server[[,]] failures and network and target partition failures or outages.

Beginning at the bottom of page 19 and continuing through the top of page 20, line 2, change the paragraph to read as follows:

FIG. 2A begins at step 2000 and runs periodically. An analysis is run every 15 minutes, or earlier if the amount of data collected crosses a threshold before 15 minutes. During each analysis period, pair-wise calculations are done on methods to determine as to which methods are affected by other methods (Block 2001). For example, deploying analysis conditioning for methods A, B and C result in pairs AB, BA, AC, CA, BC and CB being analyzed.

At page 25, lines 22-25, change the paragraph to read as follows:

FIG. 3 is a flowchart, which illustrates the optimization function of the Component Balancer. Queue entries are processed in real time as they come in from the target partitions 1008 of Fig. 1 as in the step of [[(]]Block 3001[[)]] of Fig. 3. As new groups or methods are encountered in the queue entries, they are added to the working set in memory in the Component Balancer server (Block step 3002). The working set contains the information necessary to do the optimization - such as running average response times and calls per second, etc.

At page 25, lines 30-32, continuing through the top of page 26, lines 5-10, change the paragraph to read as follows:

Optimization calculations are done every few seconds on each group in the working set (Block step 3003). A delay increment is calculated using fuzzy logic for each method targeted for optimization in the group (Block step 3004). The delay increment can be positive or negative. The largest delay increment for methods in the group is chosen and added to the current delay - but not allowed to exceed a maximum of 300 milliseconds or to go below zero (Block step 3005). The delay conditioner from the CB Server 1005 (FIG. 1) is given a maximum delay factor between 1 and 3 that it uses to calculate the delay it actually uses on the method - so the maximum any method can be delayed is 900 milliseconds. The Component Balancer 1004 (Fig. 1) automation uses a maximum delay of 3.

At page 26, lines 13-23, change the paragraph to read as follows:

The calculated delay is pushed out to the machines where the delayed methods are located by writing a registry key entry on those machines (Block step 3006). An inquiry is then made to check if no conditioned methods were called for a one-hour period (Diamond step 3010). If one hour elapses with no queue entries for a method (Yes to inquiry step 3010), that method and all of its information is removed from the working set (Block step 3011). If the answer to inquiry step 3010 is No, and no activity has occurred in the working group for 10 seconds, which is checked by inquiry step 3007- that is, no conditioned methods were called - the delay is set to zero (Block step 3012). The process then ends at bubble step 3013.

At page 26, lines 13-23, change the paragraph to read as follows:

The calculated delay is pushed out to the machines where the delayed methods are located by writing a registry key entry on those machines (Block step 3006). An inquiry is then made to check if no conditioned methods were called for a one-hour period (Diamond step 3010). If one hour elapses with no queue entries for a method (Yes to inquiry step 3010), that method and all of its information is removed from the working set (Block step 3011). If the answer to inquiry step 3010 is No, and no activity has occurred in the working group for 10 seconds, which is checked by inquiry step 3007- that is, no conditioned methods were called - the delay is set to zero (Block step 3012). The process then ends at bubble step 3013.